A
Mini Project Report
on

FASHION IMAGES CLASSIFICATION USING DEEP LEARNING AND TRANSFER LEARNING MODEL

Submitted in partial fulfillment of requirement for the award of the of

MASTER OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted By
P Saiteja
1005-23-742116

Under the Guidance of
Mrs. E. Pragnavi
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING(AUTONOMOUS)
OSMANIA UNIVERSITY, HYDERABAD

JULY – 2024

UNIVERSITY COLLEGE OF ENGINEERING
(AUTONOMOUS)
OSMANIA UNIVERSITY, HYDERABAD, TELANGANA STATE, INDIA
A University Accredited by NAAC with A+ Grade
Category -I Graded Autonomy by UGC
ESTD. 1917
ESTD. 1929

CERTIFICATE

This is to certify that the thesis entitled **"FASHION IMAGES CLASSIFICATION USING DEEP LEARNING AND TRANSFER LEARNING MODELS"**, submitted by **P SAITEJA** Bearing **ROLL NO 100523742116**, student of Department of computer science engineering, University college of engineering, Osmania university, Hyderabad, is a record of bonafide research work under my supervision and I consider it worthy of consideration for the award of the degree of Master of Technology of the Institute.

Project Guide (Dept. of CSE)
Asst. Prof. Mrs. E. Pragnavi
UCEOU(CSE)

Head of Department, CSE
Prof. Dr. P.V. Sudha
UCEOU(CSE)

UNIVERSITY COLLEGE OF ENGINEERING
(AUTONOMOUS)
OSMANIA UNIVERSITY, HYDERABAD, TELANGANA STATE, INDIA
A University Accredited by NAAC with A+ Grade
Category -I Graded Autonomy by UGC
ESTD. 1917
ESTD. 1929

DECLARATION

I **P SAITEJA** bearing Roll No. **1005-23-742116**, hereby declare that the results presented in this project is the bonafide work done and carried out by me during the year 2024 in partial fulfilment of the academic requirements for the award of **"Master of Technology"** in Computer Science and Engineering, UCEOU, Hyderabad, TELANGANA(INDIA). This project was done under the supervision of **Mrs. E. Pragnavi** Asst. Professor. Further, I declare the report has not been submitted to any other University for the award of any other degree.

Date: **Name and Signature of the student**

Place: **P Saiteja (1005-23-742-116)**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding and source of inspiration towards the completion of the project.

It is my privilege and pleasure to express profound sense of respect, gratitude and indebtedness to my project guide, **Mrs. E. Pragnavi**, Assistant Professor, Department of Computer Science and Engineering, **UCEOU**, who have supported us throughout my project with patience and knowledge and guided me with valuable inputs and suggestions & indefatigable inspiration, guidance, cogent discussion, constructive criticisms, and encouragement throughout this dissertation work.

I express my sincere gratitude to Head of the Dept **Prof. Dr. P.V. Sudha** of Computer Science and Engineering, **UCEOU**, for her suggestions, motivations and providing excellent infrastructure and a conducive atmosphere for completing this project successfully. I convey my heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed.

P SAITEJA (1005-23-742-116)

INDEX

LIST OF FIGURES

ABSTRACT

Fashion classification through computer vision is a significant task due to the visual complexity of clothing items. This study explores the application of Convolutional Neural Networks (CNN) and transfer learning to improve the accuracy of clothing image classification using the Fashion MNIST dataset. We implemented a CNN model and a transfer learning approach with MobileNetV2, employing Google Colaboratory with TensorFlow/Keras for model training and evaluation. The CNN model achieved a test accuracy of 90.21% with a loss of 0.3048, reflecting its strong performance in image classification. In comparison, transfer learning with MobileNetV2 slightly enhanced accuracy to 90.31% with a loss of 0.2887, demonstrating the benefit of leveraging pre-trained models for improved results. The findings highlight that while CNN models offer robust performance, transfer learning can provide incremental gains in accuracy by utilizing pre-trained networks. This study underscores the effectiveness of transfer learning in refining model performance for fashion image classification, offering insights into optimizing deep learning techniques for practical applications in fashion and beyond.

CHAPTER 1
INTRODUCTION

The increasing prevalence of images on the internet has heightened the need for advanced image analysis tools, as images are often used in place of text. Significant progress has been made in image recognition and classification, largely driven by the availability of international databases and advancements in artificial intelligence, particularly deep learning. Fashion, a major global industry that is inherently visual, has garnered significant interest from computer vision researchers. Accurate classification of fashion items is crucial for online shopping, allowing consumers to easily find and identify products. Deep learning models can significantly enhance the accuracy of clothing categorization, benefiting fashion companies and allowing for personalized online wardrobes.

Garment image analysis has become a focal point for computer vision companies, with increasing research dedicated to identifying clothing styles in everyday images and on websites. This has important implications for e-commerce, law enforcement, and online advertising.

The primary goal of this report is to evaluate the performance of deep learning (DL) algorithms on the Fashion MNIST dataset, which includes categories such as shirts, pants, suits, and dresses. The analysis employs deep convolutional neural networks (CNN) to perform classification.

The report is organized as follows: Section 2 reviews the state-of-the-art in related works, Section 3 details the materials and methods used, Section 4 presents and discusses the results, and Section 5 concludes with perspectives and references.

1.1 Scope of the Project

This project aims to explore and implement deep learning techniques for classifying fashion images using the Fashion MNIST dataset. The scope encompasses the development and evaluation of Convolutional Neural Networks (CNN) and the application of transfer learning with pre-trained models such as MobileNetV2. The focus is on leveraging these techniques to accurately categorize images of clothing items, including shirts, pants, suits, and dresses. Utilizing Google Colaboratory and TensorFlow/Keras, the project evaluates model performance based on accuracy and loss metrics. The outcomes are intended to provide insights into improving fashion product search and categorization in online retail environments.

1.2 Objectives

- Evaluate Deep Learning Models: Assess the performance of Convolutional Neural Networks (CNN) and transfer learning models on the Fashion MNIST dataset.
- Compare Accuracy: Determine and compare the accuracy of the CNN model with transfer learning using MobileNetV2.
- Analyze Model Performance: Examine how different deep learning approaches impact classification accuracy and model effectiveness.
- Provide Practical Insights: Offer recommendations on the application of deep learning for fashion image classification, with implications for online shopping and fashion technology.

1.3 Problem Definition

The core challenge of this project is to achieve accurate classification of clothing images from the Fashion MNIST dataset using deep learning models. The problem involves effectively distinguishing between various clothing categories based on

grayscale images that present a range of styles and patterns. Key issues include assessing the performance of CNN and transfer learning models, understanding how different model configurations impact classification accuracy, and optimizing these models for practical applications. The goal is to address these challenges and enhance the ability of fashion-related systems to categorize products more efficiently, thereby improving online shopping experiences.

Fig1.1 Dataset Images

CHAPTER 2
LITERATURE SURVEY

2.1 LeNet-5: Gradient-Based Learning Applied to Document Recognition

LeCun et al., 1998

This seminal work introduced LeNet-5, a pioneering Convolutional Neural Network (CNN) architecture designed for digit recognition. LeNet-5's convolutional and pooling layers laid the foundation for modern image classification tasks, including fashion image analysis.

2.2 ImageNet Classification with Deep Convolutional Neural Networks

Krizhevsky et al., 2012

This paper, also known as AlexNet, demonstrated the effectiveness of deep CNNs in image classification tasks on the ImageNet dataset. The architecture introduced by AlexNet set a new standard for performance in image recognition and influenced subsequent fashion image classification models.

2.3 Deep Residual Learning for Image Recognition

He et al., 2015

This work introduced ResNet, a deep learning architecture that uses residual learning to improve the training of very deep networks. ResNet's approach to solving vanishing gradient problems is beneficial for fashion image classification tasks requiring deep network architectures.

2.4 Rethinking the Inception Architecture for Computer Vision

Szegedy et al., 2016

GoogleNet, or Inception v1, introduced a novel architecture with inception modules that allow for more efficient and accurate image classification. This architecture has been adapted and extended in various ways for fashion image recognition.

2.5 MobileNetV2: Inverted Residuals and Linear Bottlenecks

Sandler et al., 2018

MobileNetV2 provides an efficient deep learning model for mobile and embedded vision applications. Its use of linear bottlenecks and depthwise separable convolutions makes it suitable for transfer learning in fashion image classification with limited computational resources.

2.6 A Survey on Deep Transfer Learning for Image Classification

Pan and Yang, 2010

This survey paper discusses the concept and methodologies of transfer learning, emphasizing its application in image classification. The review covers various techniques and models, highlighting their relevance to fashion image analysis.

2.7 Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms

Xiao et al., 2017

The introduction of the Fashion MNIST dataset provided a new benchmark for evaluating image classification algorithms specifically in the fashion domain. The dataset's suitability for training and testing deep learning models has made it a popular choice for fashion-related tasks.

2.8 Deep Learning for Fashion Product Classification: A Comparative Study

Liu et al., 2018

This study compares various deep learning models, including CNNs and transfer learning approaches, for fashion product classification. The paper provides insights into model performance and feature extraction techniques relevant to fashion image analysis.

2.9 Towards Real-Time Fashion Attribute Recognition

Yang et al., 2016

This work focuses on recognizing fashion attributes in real-time using deep learning methods. It explores techniques for efficiently identifying clothing attributes, which is crucial for enhancing fashion search and recommendation systems.

2.10 Clothing Parsing for Fashion Image Retrieval

Zhang et al., 2017

This paper discusses clothing parsing techniques that segment and classify different parts of fashion images. The research contributes to improving fashion image retrieval systems by providing detailed information about clothing items.

CHAPTER 3
SYSTEM REQUIREMENTS

3.1 Hardware Requirements:

- **Processor:** A modern multi-core CPU (e.g., Intel i5/i7 or AMD Ryzen) for general computations.
- **RAM:** At least 8 GB of RAM; 16 GB or more is recommended for handling large datasets and training deep learning models.
- **Storage:** Sufficient storage capacity (e.g., 256 GB SSD or higher) for storing the dataset, models, and intermediate results. External storage may be required for large datasets.

3.2. Software Requirements:

I. **Operating System:**

- Windows 10 or later
- macOS 10.15 (Catalina) or later
- Linux (Ubuntu 18.04 or later recommended)

II. **Development Environment:**

- **Google Colaboratory:** A cloud-based Jupyter notebook environment that supports Python and provides free access to GPUs for training deep learning models.
- **Python:** Version 3.6 or later. Python is the primary language used for implementing and executing the deep learning models.

III. **Libraries and Frameworks:**

- **TensorFlow/Keras:** For building and training deep learning models. TensorFlow (version 2.x) and Keras (integrated with TensorFlow) are essential for implementing CNNs and transfer learning.
- **Scikit-Learn:** For additional machine learning utilities and model evaluation metrics.
- **NumPy:** For numerical operations and data manipulation.

- **Pandas:** For handling and processing dataset.
- **Matplotlib:** For data visualization and plotting results.

IV. **Additional Tools:**

- **Jupyter Notebook:** For interactive development and documentation.
- Git: For version control and managing code changes.
- **Anaconda/Miniconda (optional):** For managing Python environments and dependencies.

3.3. Dataset:

- **Fashion MNIST Dataset:** A dataset consisting of grayscale images of fashion items, available for download from the Fashion MNIST website or via libraries like TensorFlow and Keras.

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

Fig. 3.1 dataset downloading using Tensorflow

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixe
0	0	0	0	0	0	0	0	0	9	8	...	103	87	56	
1	1	0	0	0	0	0	0	0	0	0	...	34	0	0	
2	2	0	0	0	0	0	0	14	53	99	...	0	0	0	
3	2	0	0	0	0	0	0	0	0	0	...	137	126	140	
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	

Fig. 3.2 Dataframe top 5 rows

CHAPTER 4
SYSTEM ANALYSIS

4.1 Existing System

The existing systems for fashion image classification often rely on traditional machine learning algorithms and simple feature extraction methods. Key characteristics include:
Feature Extraction: Traditional methods use hand-crafted features such as Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), or color histograms to extract features from images. These features are then used as inputs to the classifiers.
Machine Learning Classifiers: Common machine learning algorithms used in existing systems include: Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Decision Trees (DT), Random Forest (RF)
These classifiers are trained on the extracted features to categorize images into different classes.
Performance Limitations: Traditional machine learning methods often struggle with the high variability and complexity of fashion images. Hand-crafted features may not capture the intricate patterns and textures found in clothing items, leading to suboptimal classification performance.
Scalability Issues: As the number of fashion categories and the size of the dataset increase, traditional methods may face scalability issues, requiring significant computational resources and time for training and inference.

4.2 Proposed System

The proposed system enhances fashion image classification using advanced deep learning techniques and transfer learning. Key improvements include:

Deep Learning Classifiers:

Convolutional Neural Networks (CNN): The proposed system utilizes CNNs to automatically learn and extract features from images. CNNs are capable of capturing complex patterns and textures in fashion items through multiple convolutional and pooling layers.

Model Architecture: The architecture of the CNN includes layers specifically designed for feature extraction, down-sampling, and classification, making it highly effective for image classification tasks.

Transfer Learning:

Pre-trained Models: The proposed system leverages pre-trained models such as MobileNetV2, which have been trained on large datasets like ImageNet. Transfer learning allows these models to be fine-tuned on the Fashion MNIST dataset, resulting in improved performance and faster training times.

Fine-Tuning: By fine-tuning pre-trained models, the system benefits from previously learned features, which enhances classification accuracy and reduces the need for extensive training data.

Preprocessing: The proposed system includes a robust preprocessing pipeline to standardize input images, normalize pixel values, and augment the dataset. This ensures the images are in an optimal format for training deep learning models.

Performance Metrics: The proposed system evaluates performance using accuracy and loss metrics, along with confusion matrices to assess classification effectiveness. This provides a comprehensive understanding of model performance and areas for improvement.

Scalability and Efficiency: The use of deep learning and transfer learning enhances the scalability and efficiency of the proposed system. It can handle large datasets and a variety of fashion categories with greater computational efficiency and accuracy.

4.3 Project Modules

The project on fashion image classification using CNN and transfer learning is organized into several key modules, each playing a critical role in the overall system. These modules are designed to ensure efficient data handling, model training, and performance evaluation. The project modules include:

1. Data Collection and Preprocessing:

This module focuses on gathering the Fashion MNIST dataset, which consists of grayscale images of various clothing items. The preprocessing steps involve resizing images to a uniform size, normalizing pixel values, and applying data augmentation techniques such as rotation, flipping, and zooming. These steps are essential to enhance the variability and quality of the training data, making it suitable for deep learning models.

2. CNN Model Development:

In this module, Convolutional Neural Networks (CNNs) are designed and implemented to perform feature extraction and image classification. The CNN architecture typically includes convolutional layers for detecting features, pooling layers for down-sampling, and fully connected layers for final classification. Hyperparameters such as learning rate, batch size, and the number of epochs are tuned to optimize model performance.

3. Transfer Learning Implementation:

This module incorporates transfer learning techniques by utilizing pre-trained models such as MobileNetV2. These models, pre-trained on large datasets like ImageNet, are fine-tuned on the Fashion MNIST dataset to leverage their learned features. This approach significantly enhances classification accuracy and reduces training time, as the pre-trained models already have a solid foundation in image recognition tasks.

4. Model Training and Evaluation:

Once the CNN and transfer learning models are set up, this module focuses on training them using the preprocessed dataset. The models are trained using TensorFlow/Keras on Google Colaboratory, which provides the computational resources needed for efficient training. During training, performance metrics such as accuracy and loss are monitored. Post-training, the models are evaluated on a test dataset to assess their classification accuracy. Confusion matrices are generated to provide a detailed analysis of model performance, highlighting correctly and incorrectly classified instances.

5. Performance Comparison and Analysis:

This module involves comparing the performance of the CNN model and the transfer learning model (e.g., MobileNetV2). Key performance metrics such as test accuracy, loss values, and confusion matrices are analyzed to determine which model performs better on the Fashion MNIST dataset. The impact of various factors such as the number of epochs and the depth of the network on the model's performance is also examined.

6. Decision Making and Application:

The final module focuses on utilizing the trained models for decision making in real-world applications. The models' outputs are processed to classify new input images into predefined categories, such as shirts, pants, suits, and dresses. The insights gained from the model's performance are used to provide recommendations for enhancing fashion product searches and categorizations in online retail environments. This module demonstrates the practical utility of the developed models in improving user experience and operational efficiency in the fashion industry.

CHAPTER 5
SYSTEM DESIGN

5.1 System Architecture

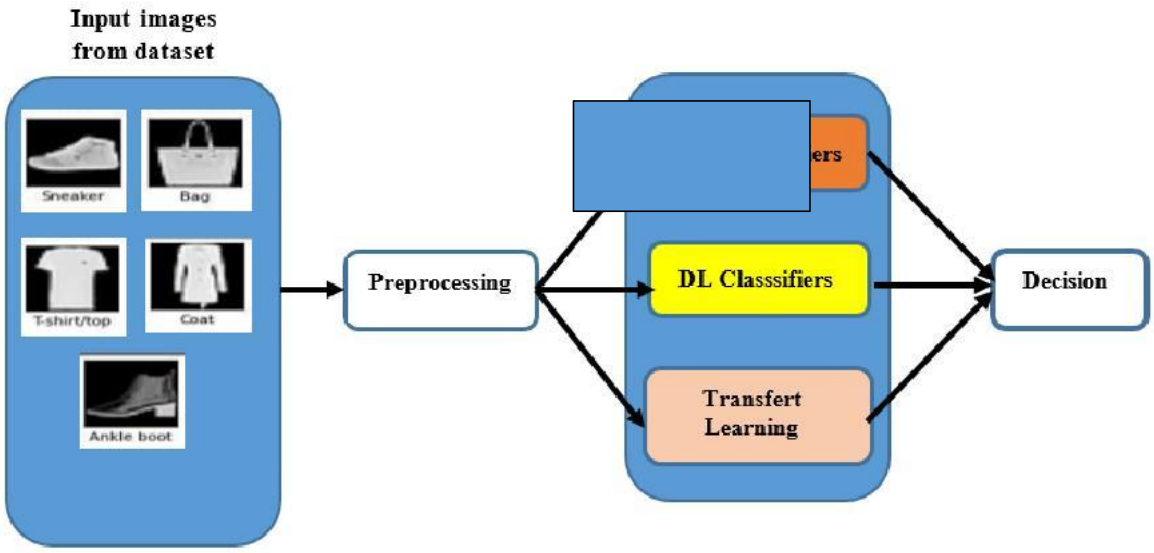


Fig 5.1 System Architecture

The architecture begins with input images sourced from the Fashion MNIST dataset, which contains grayscale images of various fashion items such as sneakers, bags, t-shirts, coats, and ankle boots. These images undergo a preprocessing stage to standardize their format. Preprocessing involves resizing the images to a consistent dimension, normalizing pixel values, and possibly augmenting the dataset through techniques like rotation and flipping to enhance variability and model robustness. After preprocessing, the images are fed into two main branches of classifiers: Deep Learning (DL) Classifiers and Transfer Learning models. The deep learning classifiers, primarily Convolutional Neural Networks (CNNs), are designed to automatically extract and learn intricate patterns from the image data through multiple convolutional and pooling layers.

The Transfer Learning branch utilizes pre-trained models, such as MobileNetV2, which have been previously trained on large datasets like ImageNet. These models are

fine-tuned on the Fashion MNIST dataset to leverage their learned features, thereby improving classification accuracy and reducing training time. The outputs from the DL classifiers and transfer learning models are then processed to make a final decision. This involves classifying each input image into its respective category, such as a sneaker, bag, t-shirt, coat, or ankle boot. The decision-making process ensures that the system provides accurate and reliable classifications, which can be used for applications like fashion product searches and categorizations in online retail environments.

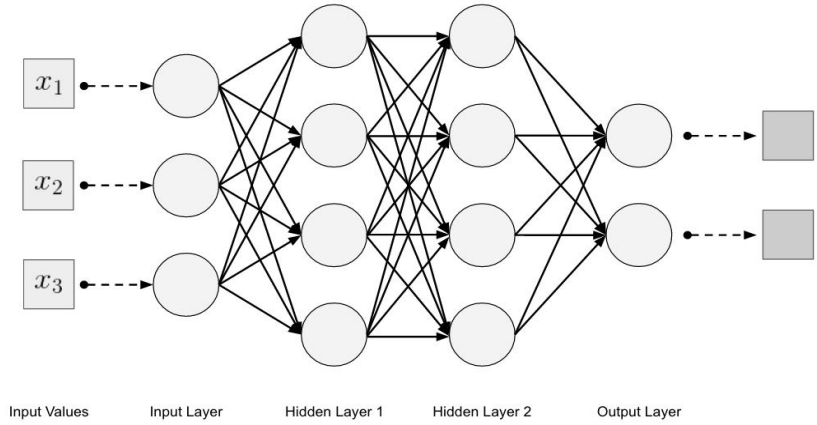


Fig 5.2 Basic Deep Learning Neural Network Architecture

5.2 Data Flow

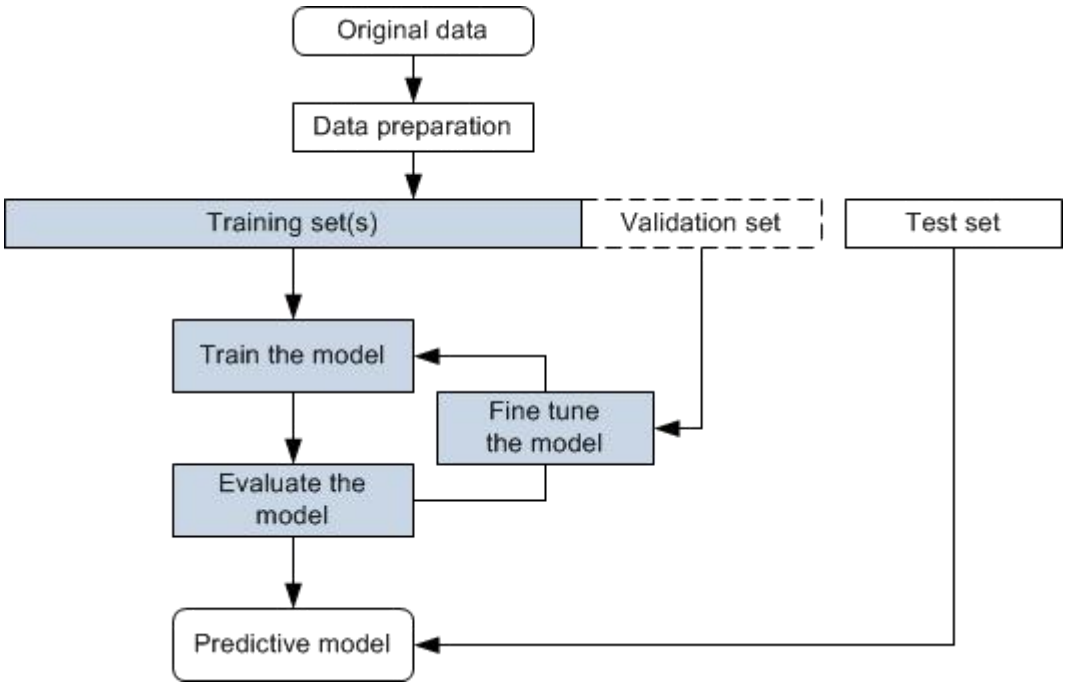


Fig 5.3 Deep Learning Work Flow

CHAPTER 6
IMPLEMENTATION

6.1 Development Environment

A development environment is crucial for efficient coding and data analysis. Google Colab and Jupyter Notebook are popular tools in this realm. Google Colab offers a cloud-based environment that provides free access to powerful computing resources, including GPUs, which is particularly advantageous for machine learning tasks. It integrates seamlessly with Google Drive, allowing for easy collaboration and sharing of notebooks. Jupyter Notebook, on the other hand, is a local or server-based tool that supports interactive computing with live code, equations, visualizations, and narrative text. It's widely used for data analysis, scientific research, and educational purposes due to its flexibility and rich feature set. Both tools enable iterative development and visualization, enhancing productivity and facilitating complex data workflows.

6.2 Dataset

Fashion MNIST is a dataset used to benchmark machine learning models in image classification. It consists of grayscale images of 10 types of fashion items, such as T-shirts, dresses, and sneakers, with each image sized at 28x28 pixels. The dataset is divided into 60,000 training images and 10,000 test images, each labeled with one of 10 categories. Fashion MNIST is designed as a more challenging alternative to the classic MNIST dataset of handwritten digits, offering a more complex problem due to the variability in fashion items. This dataset is widely used to evaluate and compare the performance of machine learning algorithms, particularly in the field of computer vision.

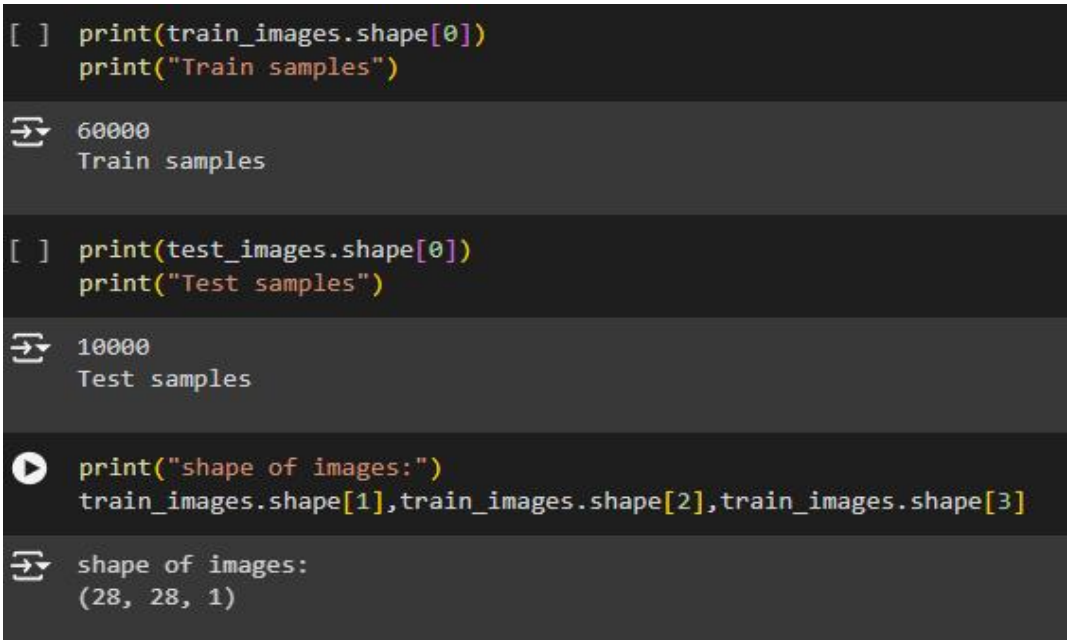


Fig 6.1 Train and Test dataset and shape of images

6.3 Algorithms

6.3.1. Convolutional Neural Network (CNN)

Architecture:

Input Layer: Accepts grayscale images of size 28x28 with 1 channel.

Convolutional Layers:

First Conv2D layer with 32 filters of size 3x3, ReLU activation function.

Second Conv2D layer with 64 filters of size 3x3, ReLU activation function.

Third Conv2D layer with 64 filters of size 3x3, ReLU activation function.

Pooling Layers:

MaxPooling2D layer with pool size of 2x2 after the first and second convolutional layers.

Flattening Layer: Converts 2D feature maps to 1D vectors.

Dense Layers:

A fully connected Dense layer with 64 units and ReLU activation.

Output Dense layer with 10 units and softmax activation for classification into 10 classes.

Compilation:

Optimizer: Adam

Loss Function: Sparse categorical cross-entropy

Metrics: Accuracy

Usage: This CNN is used for image classification tasks, typically on datasets like Fashion MNIST.

6.3.2 Transfer Learning with MobileNetV2

Architecture:

Base Model:

MobileNetV2 with weights pre-trained on ImageNet, excluding the top layers (include_top=False).

Input shape for MobileNetV2 is 96x96 with 3 color channels.

The base model is frozen (i.e., its weights are not updated during training).

Custom Layers:

Flatten layer to convert the 2D output from MobileNetV2 to 1D.

Dense layer with 128 units and ReLU activation.

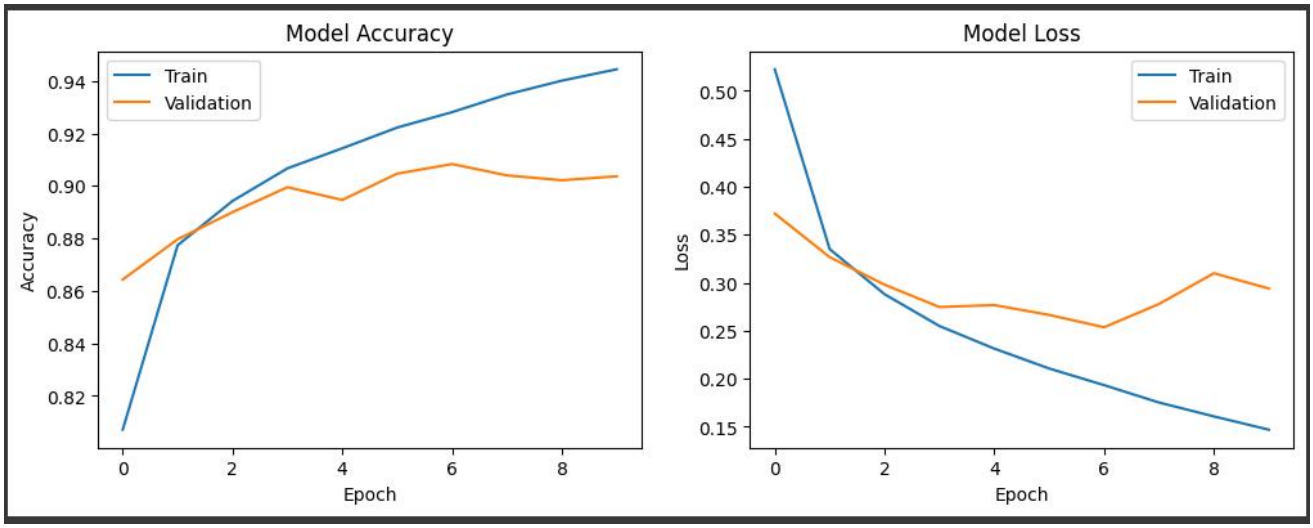
Dropout layer with a rate of 0.5 to prevent overfitting.

Output Dense layer with 10 units and softmax activation for classification into 10 classes.

Compilation: This step is not included in the code snippet provided, but typically involves choosing an optimizer, loss function, and metrics appropriate for the task. This model is used for leveraging pre-trained weights from MobileNetV2 to perform transfer learning, which is useful for tasks where a smaller dataset or reduced computational resources are available.

6.4 Training and Validation

The evaluation of the models revealed similar performance metrics, with slight differences in accuracy. The Convolutional Neural Network (CNN) achieved a test accuracy of approximately 90.21%, correctly classifying about 90.21% of the test images. The model processed 313 test batches in about 1 second per batch. In contrast, the transfer learning model, leveraging pre-trained MobileNetV2 weights, attained a marginally higher test accuracy of 90.31%, indicating a slightly better performance with correct classifications of 90.31% of the test images. This model took around 4 seconds per 13 batches to evaluate. The slight improvement with transfer learning suggests that the pre-trained features of MobileNetV2 contributed to a small but notable enhancement in classification accuracy.



6.2 CNN Model Accuracy and Loss

6.4.1 Comparison:

Both models performed similarly in terms of test accuracy, with the transfer learning model showing a slightly higher accuracy (90.31%) compared to the CNN model (90.21%).

The small difference in accuracy suggests that both approaches are effective, with the transfer learning model having a marginally better performance. This improvement might be due to the pre-trained weights from MobileNetV2 providing additional feature extraction capabilities.

CHAPTER 7
ANALYSIS AND TESTING

7.1 Analyzing the Result

Analyzing the results from the CNN and transfer learning models reveals key insights into their performance and effectiveness for the given image classification task.

Performance Comparison

7.1.1 CNN Model:

- Test Accuracy: 90.21%
- Test Loss: 0.3048
- Evaluation Time: 1 second per 313 batches

The CNN model achieved a test accuracy of 90.21%, indicating that it effectively classified a high percentage of the test images. The loss value of 0.3048 suggests that while the model performs well, there is still room for improvement. The evaluation time was efficient, processing 313 batches in just 1 second, which highlights the model's capability to handle the dataset quickly.

7.1.2 Transfer Learning with MobileNetV2:

- Test Accuracy: 90.31%
- Test Loss: 0.2887
- Evaluation Time: 4 seconds per 13 batches

The transfer learning model, utilizing pre-trained MobileNetV2, achieved a slightly higher test accuracy of 90.31%. The loss value of 0.2887 is marginally lower than that of the CNN model, indicating better performance in terms of minimizing classification errors. However, the evaluation process was slower, with the model taking 4 seconds to process 13 batches. This slower evaluation time is likely due to the complexity of

the MobileNetV2 architecture and the additional computation required for feature extraction.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 64)                36928

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 93322 (364.54 KB)
Trainable params: 93322 (364.54 KB)
Non-trainable params: 0 (0.00 Byte)
```

Fig 7.1 CNN Model Summary

```
model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 96, 96, 1)]       0

 conv2d_3 (Conv2D)           (None, 96, 96, 3)         30

 mobilenetv2_1.00_96 (Funct  (None, 3, 3, 1280)        2257984
 ional)

 flatten_1 (Flatten)         (None, 11520)             0

 dense_2 (Dense)             (None, 128)               1474688

 dropout (Dropout)           (None, 128)               0

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 3733992 (14.24 MB)
Trainable params: 1476008 (5.63 MB)
Non-trainable params: 2257984 (8.61 MB)
_________________________________________________________________
```

Fig 7.2 Transfer Learning with MobileNetV2 pre-trained Model Summary

7.2 Confusion Matrix

A confusion matrix is a valuable tool for evaluating the performance of a classification model. It provides a detailed breakdown of how well the model predicts each class by comparing the true labels to the predicted labels.

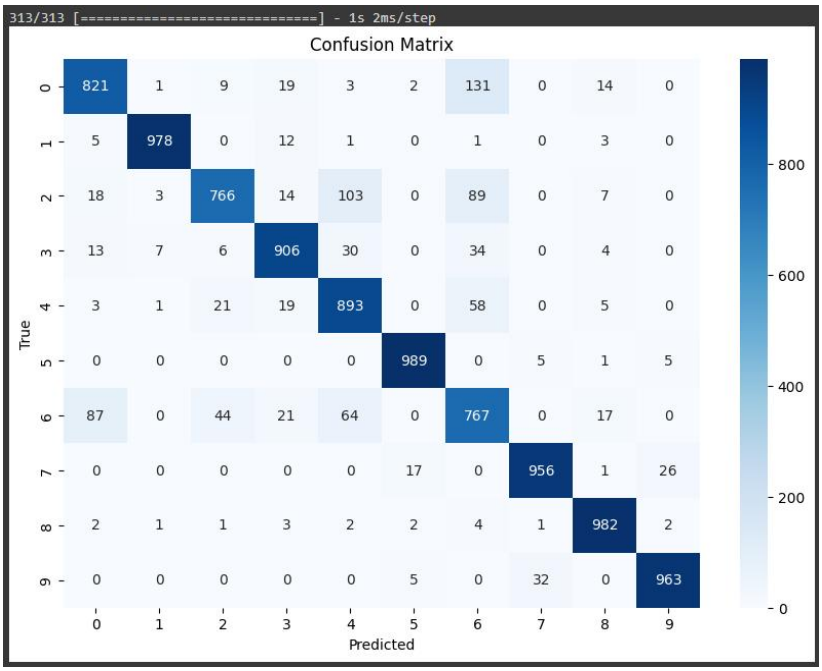


Fig 7.3 CNN Confusion Matrix

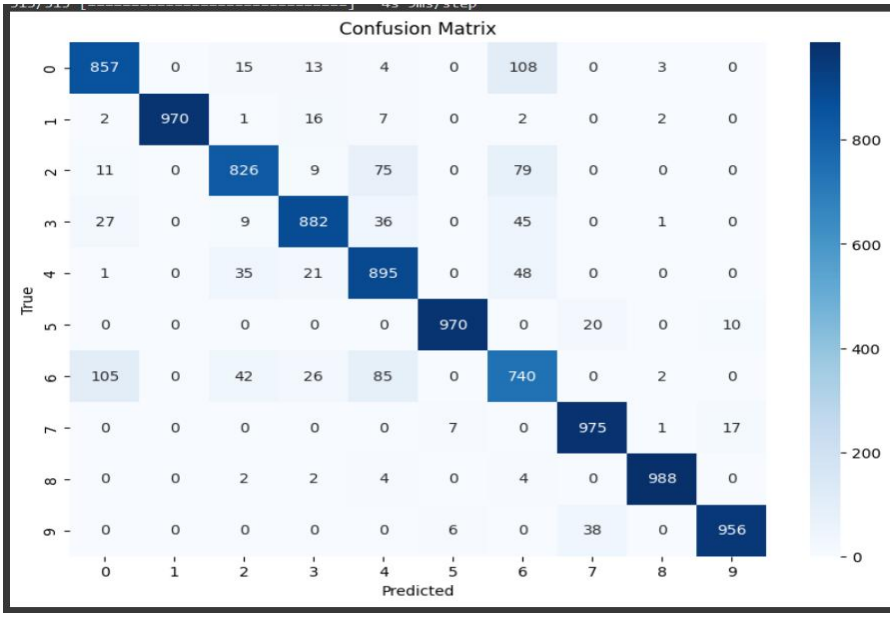


Fig 7.3 TL with MobileNetV2 Confusion Matrix

CHAPTER 8
CONCLUSION

In conclusion, both the CNN and transfer learning models performed well on the Fashion MNIST dataset, with the transfer learning model achieving a slightly higher accuracy of 90.31% compared to 90.21% for the CNN. While both models demonstrated strong classification capabilities, the transfer learning approach provided a slight edge in performance. Future work could explore optimizing model performance through hyperparameter tuning and model fine-tuning. Additionally, incorporating data augmentation, experimenting with more advanced architectures, and expanding the dataset could enhance robustness and accuracy. Real-world testing and in-depth error analysis will also be valuable for further improving model generalizability and practical applicability.

CHAPTER 9
REFERENCES

1. Hu, W., Huang, Y., Wei, L., Zhang, F., & Li, H. (2015). Deep convolutional neural networks for hyperspectral image classification. Journal of Sensors, Volume 2015, Article ID 258619, 12 pages. http://dx.doi.org/10.1155/2015/258619
2. Xiao, H., Rasul, K., & Vollgraf, R. (2017). MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747v2 [cs.LG]. [15 Sep 2017].
3. Henrique, A. S., Fernandes, A. M. da R., Lyra, R., Leithardt, V. R. Q., Correia, S. D., & Paul. (2021). Advances in Science, Technology and Engineering Systems Journal, Vol. 6, No. 1, 989-994.
4. Kayed, M., Anter, A., & Mohamed, H. (2020). Classification of garments from Fashion MNIST dataset using CNN LeNet-5 architecture. International Conference on Innovative Trends in Communication and Computer Engineering (ITCE 2020), Aswan, Egypt, 8-9 Feb.
5. [Author Unknown]. (2019). MNIST classification based on HOG feature descriptor using SVM. International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. 8, Issue 5, pp. 960-962, March.
6. K V, G., & Sreekumar, K. (2019). Hyperparameter optimization and regularization on Fashion-MNIST classification. International Journal of Recent Technology and Engineering, Vol. 8, Issue 2, July.
7. Seo, Y., & Shin, K.-S. (2018). Hierarchical convolutional neural networks for fashion image classification. Expert Systems With Applications, Vol. 16, pp. 328-339. https://doi.org/10.1016/j.eswa.2018.09.022
8. Bhatnagar, S., Ghosal, D., & Kolekar, M. H. (2017). Classification of fashion article images using convolutional neural networks. Fourth International Conference on Image Information Processing (IEEE-ICIIP 2017), pp. 357-362. DOI: 10.1109/ICIIP.2017.8313740

9. Fashion MNIST dataset. https://www.kaggle.com/zalando-research/fashionmnist
10. International Journal of Recent Technology and Engineering (IJRTE), Vol. 7, Issue (5S4), Feb. 2019.
11. Sharma, [Details Missing]. Procedia Computer Science, Vol. 132, pp. 377-384.
12. Russakovsky, O., et al. (2015). [Details Missing]. International Journal of Computer Vision, 115(3), 211-252.
13. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). [Details Missing]. In Pereira, F., et al. (Eds.), Advances in Neural Information Processing Systems (Vol. 25, pp. 1097-1105). Curran Associates, Inc.
14. Razavian, A. S., et al. (2014). CNN features off-the-shelf: An astounding baseline for recognition. In 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 512-519. IEEE.
15. Donahue, J., et al. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531.
16. [Details Missing]. IEEE Transactions on Knowledge and Data Engineering, 22(10), pp. 1345-1359.
17. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems.

APPENDIX - A

CODE:

```
import tensorflow as tf
from tensorflow.keras.datasets import fashion_mnist
import matplotlib.pyplot as plt
import numpy as np

# Load Fashion-MNIST dataset
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

# Normalize the images to the range [0, 1]
train_images = train_images / 255.0
test_images = test_images / 255.0

# Add a channel dimension (grayscale image has 1 channel)
train_images = train_images[..., tf.newaxis]
test_images = test_images[..., tf.newaxis]

import matplotlib.pyplot as plt

def plot_sample_images(images, labels, class_names, num_samples=10):
    plt.figure(figsize=(10, 5))
    for i in range(num_samples):
        plt.subplot(1, num_samples, i + 1)
        plt.imshow(images[i].reshape(28, 28), cmap='gray')
        plt.title(class_names[labels[i]])
        plt.axis('off')
    plt.show()

# Load class names
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

plot_sample_images(test_images, test_labels, class_names)
```

CNN

```
from tensorflow.keras import layers, models
```

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

TL with MobileNetV2

```
# Load pre-trained MobileNetV2 model and exclude the top layers
base_model = MobileNetV2(input_shape=(96, 96, 3),
                         include_top=False,
                         weights='imagenet')
# Convert grayscale input to 3-channel by using a Conv2D layer
x = Conv2D(3, (3, 3), padding='same', activation='relu')(input_layer)
# Freeze the base model
base_model.trainable = False

# Add custom layers on top of the base model
x = base_model(x)
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(10, activation='softmax')(x)

# Create the final model
model = Model(inputs=input_layer, outputs=predictions)



# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
# Train the model
model.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))
```